***// DEPTH FIRST SEARCH***

#include <stdio.h>

#include <stdlib.h>

*/\* DFS function calls dfs on node v and keeps track of the component number through componentId parameter. \*/*

void dfs(int v, int\*\* graph, int n, int\* vertexBelongsTo, int componentId);

*/\* Components returns the number of components in the graph and updates vertexBelongsTo such that ith index of vertexBelongsTo is the component to which ith vertex belongs. Make sure that the number of the ith component is as minimum as possible, i.e. if there are 3 components {1,4,5} and {2,3} and {6,7}, then {1,4,5} component number is 1, that of {2,3} is 2, and that of {6,7} is 3. \*/*

int components(int \*\*graph, int n, int \*vertexBelongsTo);

int main() {

int t;

scanf("%d", &t);

while(t--) {

int n;

scanf("%d", &n);

int \*\*graph = (int \*\*) malloc(sizeof(int \*) \* n);

for(int i = 0; i < n; i++) {

graph[i] = (int \*) malloc(sizeof(int) \* n);

for(int j = 0; j < n; j++) {

scanf("%d", &graph[i][j]);

}

}

int \*vertexBelongsTo = (int \*) malloc(sizeof(int) \* n);

int numberOfComponents = components(graph, n, vertexBelongsTo);

printf("%d\n", numberOfComponents);

for(int i = 0; i < n; i++) {

printf("%d ", vertexBelongsTo[i]);

}

printf("\n");

}

return 0;

}

int components(int \*\*graph, int n, int \*vertexBelongsTo) {

int compId = 1;

for(int i = 0; i < n; i++)

{

vertexBelongsTo[i] = 0;

}

for(int i = 0; i < n; i++)

{

if(vertexBelongsTo[i] == 0)

{

dfs(i, graph, n, vertexBelongsTo, compId);

compId++;

}

}

return compId - 1;

}

void dfs(int v, int \*\*graph, int n, int \*vertexBelongsTo, int componentId) {

vertexBelongsTo[v] = componentId;

for(int j = 0; j < n; j++)

{

if(graph[v][j] == 1 && vertexBelongsTo[j] == 0)

{

vertexBelongsTo[j] = componentId;

dfs(j, graph, n, vertexBelongsTo, componentId);

}

}

}